

A Comparison of Techniques for Scheduling Earth-Observing Satellites

Al Globus
CSC at NASA Ames

James Crawford
NASA Ames

Jason Lohn
NASA Ames

Anna Pryor
NASA Ames

Abstract

Scheduling observations by coordinated fleets of Earth Observing Satellites (EOS) involves large search spaces, complex constraints and poorly understood bottlenecks, conditions where evolutionary and related algorithms are often effective. However, there are many such algorithms and the best one to use is not clear. Here we compare multiple variants of the genetic algorithm, stochastic hill climbing, simulated annealing, squeaky wheel optimization and iterated sampling on ten realistically-sized EOS scheduling problems. Schedules are represented by a permutation (non-temperal ordering) of the observation requests. A simple deterministic scheduler assigns times and resources to each observation request in the order indicated by the permutation, discarding those that violate the constraints created by previously scheduled observations. Simulated annealing performs best. Random mutation outperform a more 'intelligent' mutator. Furthermore, the best mutator, by a small margin, was a novel approach we call 'temperature dependent random swap' that makes large changes in the early stages of evolution and smaller changes towards the end of search.

Introduction

Approximately 60 scientific and commercial Earth Observing Satellites (EOS) circle the globe collecting images. Nearly all of these satellites are chronically oversubscribed, i.e., there are far more observation requests than can possibly be satisfied. Scheduling systems are used to satisfy as many of these requests as possible, favoring higher priority requests. The EOS observation scheduling problem is also characterized multiple complex constraints, including power, thermal, data capacity, and the limited time each satellite spends over each target. Some EOS satellites can make hundreds of observations per day, each observation may have dozens of short, fixed, predictable imaging opportunities a week, and request backlogs often number in the thousands. Thus, finding an optimal or near-optimal schedule involves searching a very large space. In general, the size of the search space precludes a complete search requiring some form of random, albeit directed, search.

EOS scheduling is an example of an oversubscription

scheduling problems where there are more requests for a resource than can be satisfied, insuring that some requests remain unfulfilled. Such problems also include scheduling planetary probes, telescopes, the deep space network, wind tunnels and other test facilities. These problems involve allocation of expensive resources, often with complex constraints and prioritized tasks. Poor schedules will result in under-utilization of expensive facilities and thus a substantial waste of money, often taxpayer dollars. There are a number of scheduling algorithms that address oversubscription problems in general or EOS scheduling in particular, but few systematic comparisons between them.

Our study compares thirteen scheduling algorithms including variants of stochastic hill climbing (Baluja 1995), simulated annealing (Kirkpatrick, Gelatt, & AndVecchi 1983), genetic algorithms (GA) (Holland 1975), squeaky wheel optimization (Joslin & Clements 1999), and iterated sampling (ISAMP) (Crawford & Baker 1994). Hill climbing and simulated annealing mutate a single schedule looking for an optimum, the difference being that hill climbing is strictly greedy whereas simulated annealing will occasionally take backwards steps attempting to avoid local optima. We test both steady-state and generational genetic algorithms, both population-based techniques. Random mutation and crossover operators are used as well as 'squeaky' mutators which try to make intelligent changes to the schedule. One of the mutation operators, temperature dependent random swap described below, is novel. It marginally outperformed purely random mutation. ISAMP is effectively random search.

Evolutionary algorithms, particularly the genetic algorithm (GA), have been used to schedule a wide variety of tasks. For example, Syswerda and Palmucci scheduled the U.S. Navy's System Integration Test Station laboratory for F-14 jet fighters using a GA with a permutation of tasks representation and a fast greedy scheduler to place tasks, one at a time, in the schedule (Syswerda & Palmucci 1991). This work motivated our research. Philip Husbands provides a good, if somewhat dated, survey of GA for scheduling problems (Husbands 1994).

Computational scheduling techniques have been ap-

plied to the EOS scheduling problem by several authors, including:

1. Sherwood (Sherwood *et al.* 1998) used ASPEN, a general purpose scheduling system, to automate scheduling of NASA's EO-1 satellite.
2. Potter and Gasch (Potter & Gasch 1998) described a clever algorithm for scheduling NASA's Landsat 7 satellite featuring greedy search forward in time with fixup to free resources for high priority observation.
3. Lamaitre's group has examined EOS scheduling issues including comparison of multiple techniques. See, for example, (Lamaitre, Verfaillie, & Bataille 1998), (Bensana, Lemaitre, & Verfaillie 1999) and (Lamaitre *et al.* 2000).
4. Wolfe and Sorensen (Wolfe & Sorensen 2000) compared three algorithms on the window-constrained packing problem, which is related to EOS scheduling. They found that the genetic algorithm produced the best schedules, albeit at a significant CPU cost.
5. Our group has published earlier results in (Globus *et al.* 2002) and (Globus *et al.* 2003).

In the next section we describe the scheduling problem and our model. A description of the scheduling techniques follows. The nature and results of our computational experiments are then presented along with a discussion of results and conclusions.

EOS Scheduling Problem

We first describe the real EOS scheduling problem. Then we describe the model problems used in this experiment.

EOS scheduling attempts to take as many high-priority observations as possible within a fixed period of time with a fixed set of satellite-borne sensors. For example, the Landsat 7 satellite scheduler is considered to have done a good job if 250 observations are made each day. EOS scheduling is complicated by a number of important constraints. Potin (Potin 1998) lists some of these as:

1. Revisit limitations. Observation targets must be within sight of the satellite. EOS satellites travel in fixed orbits, usually about 800 km up and 100 minutes per orbit. These orbits pass over any particular place on Earth at limited times so there are only a few observation windows (and sometimes none) for a given target within a given time period.
2. Time required to take each image. Most Earth observing satellites take a one-dimensional image and use the spacecraft's orbital motion to sweep out the area to be imaged. For example, a Landsat image requires 24 seconds of orbital motion.
3. Limited on-board data storage. Images are typically stored on a solid state recorder (SSR) until they can be sent to the ground.

4. Ground station availability. The data in the SSR is sent to the ground (SSR dumps) when the satellite passes over a ground station. Ground station windows are limited as with any other target.
5. Transition time between look angles (slewing). Some instruments are mounted on motors that can point side-to-side (cross-track). These motors can wear out so slewing should be minimized.
6. Pointing angle. The highest resolution images are taken when the target is directly below the satellite (nadir pointing). Other pointing angles are sometimes required for special purposes.
7. Power availability. Most satellites have very restrictive power budgets.
8. Thermal control. As satellites pass in and out of the Earth's shadow the thermal environment changes radically. This places constraints on sensor use.
9. Coordination of multiple satellites. In particular, assigning image collection responsibility appropriately.
10. Cloud cover. Some sensors cannot see through clouds.
11. Stereo pair acquisition or multiple observations of the same target by different sensors or the same sensor at different times.

To make confident statements about the best method to solve a set of real-world problems it is not enough to solve random related problems. Watson and his collaborators at Colorado State University (Watson *et al.* 1999) examined the performance of a number of algorithms on model job-shop problems. They found that although sophisticated algorithms performed very well on random problems, they did poorly if the problems were modified to exhibit structure based on real-world problems. Simple algorithms performed better on the more realistic problems. To take this effect into account, although our model problems contain randomly generated observation targets, these are limited to land areas, the satellites are in realistic orbits, and our model problems exhibit all of Potin's constraints except the last two.

Table 1 contains the variable parameters of the model problems. Each problem consist of one to three satellites in Sun-synchronous orbits (orbits in which the equator is crossed at the same local time each orbit) for one week. Multiple satellites are spaced ten minutes apart along the same orbit. Each satellite carries one sensor mounted on a cross-track slewable motor that can point up to 24 degrees to either side of nadir (nadir is straight down).

Each problem is assigned 2100 *NumberOfSatellites* observation targets randomly generated on land. Each target is assumed to lie in the center of a rectangle who's size depends on the observing time. Not all of these targets will be visible during the one week period. The longer the required observation time the fewer targets will be observable. Any satellite is allowed to make any

problem name	satellites			observations				weights		
	number	SSR	slew ($^{\circ}$ /sec)	number	time (sec)	SSR use	priority	w_p	w_s	w_a
1	1	75	2	1934	24	1	1-6,50	1	0.01	0.02
2	3	50	2	6041	36	1,3,5	1-6,50	1	0.01	0.50
3	3	50	2	6114	24	1,3,5	1-6,50	1	0.01	0.00137
4	3	75	2	6114	24	1,3,5	1-6,50	1	0.01	0.02
5	3	50	10	6041	36	1,3,5	1-51,5	1	0.5	0.02
6	3	75	2	6114	24	1,5,8	1-16,50	1	0.10	0.20
7	3	75	2	5465	48	1,5,8	1-16,50	1	0.10	0.20
8	3	75	2	5465	48	1,3,5	1-6,50	1	0.01	0.02
9	2	75	2	3995	24	1,3	1-6,50	1	0.01	0.02
10	3	100	1	6041	36	1,10,25	1-6,50	1	0.1	0.7

Table 1: Parameters of the EOS scheduling problems tested. The first column is the problem name. The next three columns relate to the satellites, the number of satellites, the size of the SSR, and the slew rate. The next four columns relate to the observations, the number of observation targets the satellites could see, the time necessary for each observation, the SSR used by each observation and the priority of the observations. Where there is more than one number in the SSR column the values are evenly divided among the observation targets. The format of the priority column is lowest-highest, numberOfLevels. 'numberOfLevels' is the number of distinct priority levels divided evenly among the observations. The last three columns are the weights used in the fitness function (see equation 1).

observation. Each observation target counted in table 1 is within view of a satellite at least once, usually several times and sometimes over 20. Orbits and observation windows were determined by the free version of the Analytical Graphics Inc.'s Satellite Tool Kit, also known as the STK (see www.stk.com). The STK uses highly accurate orbital determination methods.

There is one ground station in Alaska. Whenever a satellite comes within sight of the ground station it is assumed to completely empty its SSR, which is then available for additional observation storage. There are approximately 75 SSR dumps per spacecraft during a week. Since some orbits are over oceans and all targets are on land, some SSR dump opportunities are wasted on an empty, or nearly empty, SSR. We model power and thermal constraints using so-called duty cycle constraints, the approach taken by Landsat 7. A duty cycle constraint requires that the sensor not be turned on for longer than a maximum time within any interval of a certain length. Our model problem uses the Landsat 7 duty cycles. Specifically, a sensor may not be used for more than:

1. 34 minutes in any 100 minute period,
2. 52 minutes in any 200 minute period, or
3. 131 minutes in any 600 minute period.

The fitness (quality) of each schedule is determined by a weighted sum (smaller values indicate better fitness):

$$F = w_p \sum_{O_u} P_o + w_s S + w_a A \quad (1)$$

where F is the fitness, O_u is the set of unscheduled observation, P_o is an observation's priority, S is the mean time spent slewing per scheduled observation, A

is the mean off-nadir pointing angle for all scheduled observations, w stands for weight.

Scheduling Algorithms

This study compares thirteen search algorithms applied to ten EOS scheduling problems. The search techniques were hill climbing, simulated annealing, steady state and generational genetic algorithms, and ISAMP (essentially random search). By using a more intelligent mutation operator, these algorithms (except ISAMP) become variants of squeaky wheel optimization (Joslin & Clements 1999). In squeaky wheel optimization, those observations that are not scheduled are given a high likelihood of being involved in mutation.

Our work focuses on *permutation-based* (Syswerda & Palmucci 1991) approaches to scheduling problems. The key insight underlying such approaches is that if we could greedily schedule EOS observation requests in an optimal order then we would produce an optimal schedule.¹ Thus, a greedy scheduler allows us to search the space of permutations rather than the space of schedules. This change of representation has two key advantages: First, and most important, the greedy scheduler can take any permutation and produce a feasible (though generally sub-optimal) schedule. This means that mutation and crossover operations never stray into infeasible space. This is in contrast to methods that search in the space of schedules. These must work hard to maintain feasibility, or find ways to assign infeasible schedules a fitness that guides evolution (i.e., infeasible schedules cannot all be assigned the

¹We should note that proving optimality for a permutation-based method requires a detailed analysis of the constraints and optimization criteria of the domain as well as the details of the greedy scheduler.

same fitness). Second, if there are many possible times at which observations can be scheduled it is often the case that the space of possible permutations is significantly smaller than the space of possible schedules.

Thus, we represent a schedule as a permutation or arbitrary, non-temporal ordering of the observations. The observations are scheduled one at a time in the order indicated by the permutation. In psuedo-code:

```
int[] permutation = permute 1-numObservations
for(int i = 1; i <= numObservations; i++)
    if (observation at permutation[i]
        not violate current constraints)
        schedule observation at permutation[i]
```

A simple, greedy, deterministic scheduler assigns resources to observations in the order indicated by the permutation. This produces a set of timelines with all of the scheduled observations, the time they were taken, and the resources (SSR, sensor, slew motor setting) used. The greedy scheduler assigns times and resources to observations using earliest-first scheduling heuristics without violating constraints. If an observation cannot be scheduled without violating the current constraints (those created by scheduling observations from earlier in the permutation), the observation is left unscheduled.

The first greedy scheduler we implemented employed earliest-first heuristics starting at *time* = 0. For observations that could be taken at several different times (windows) the earliest window that did not violate current constraints was chosen. We discovered that better schedules are generated if, for each observation, 'earliest-first' starts at some initial time chosen by evolution rather than *time* = 0.

The initial time, set randomly at first, is generally different for each observation. The greedy scheduler starts at the initial time and looks forward for a constraint-free window where the observation can be scheduled. If none is found before the end of time, the scheduler wraps around to *time* = 0 and continues. The time each observation is scheduled (or, if unscheduled, what time 'earliest-first' search started) is stored and preserved by mutation and crossover. The extra scheduling flexibility may explain why this approach works better than earliest-first starting at *time* = 0.

Constraints are enforced by representing sensors, slew-motors and SSRs as timelines. Scheduling an observation causes timelines to take on appropriate values (i.e., in use for a sensor, slew motor setting, amount of SSR memory available) at appropriate times. These timelines are checked for constraint violations as the greedy scheduler attempts to schedule additional observations.

The simplest search technique tested was ISAMP, which is essentially a random search. With ISAMP, each schedule is generated from a random permutation with random start times for the greedy scheduler. The rest of the search techniques start with random permutations and generate new permutations with mutation and/or crossover. The techniques tested were:

1. Hill climbing (Hc)², which starts with a single randomly generated permutation. This permutation (the parent) is mutated to produce one new permutation (a child) which, if the child represents a more fit schedule than the parent, replaces the parent.
2. Simulated annealing (Sa), which is similar to hill climbing except that less fit children replace the parent with probability $p = e^{-\frac{\Delta F}{T}}$ where ΔF is the fitness and T is an artificial temperature. The temperature starts at 100 (arbitrary units) and is multiplied by 0.92 every 1000 children (100,000 children are generated per job).
3. A steady-state tournament selection genetic algorithm (Gs), which is similar to hill climbing except that parents are chosen from a population of 100 schedules. Parents are chosen by a tournament where the most fit of two randomly selected schedules becomes the parent. Children replace one member of the population, chosen by randomly selecting two schedules and replacing the least fit.
4. A generational elitist genetic algorithm (Gg), which is identical to Gs except that each generation, the best 10 schedules are copied into a second population. Then another 100 schedules are generated and placed in the new population choosing parents with the same tournament selection method. Once the new population is complete, the old is discarded and the process repeated.

Each search technique (except ISAMP) was tested with three mutation operators:

1. Random swap (Sr)³. Two permutation locations are chosen at random and the observations are swapped, with 1-15 swaps (chosen at random) per mutation. Earlier experiments (Globus *et al.* 2003) determined that allowing more than one swap improved scheduling.
2. Temperature-dependent swap (Td). Here the number of swaps (1-15) is still chosen at random but with a bias. Early in evolution a larger number of swaps tend to be used, and later in evolution fewer swaps are performed. This is analogous to the 'temperature' dependent behavior of simulated annealing. The choice of the number of swaps is determined by a weighted roulette wheel where the weights vary linearly as evolution proceeds. Weights start at n and end at $16 - n$ where n is the number of swaps. In the beginning of search temperature dependent swap allows large jumps to find a deep local well. Near the end of search the schedule is close to the minima and the smaller mutations are more likely to move downhill. This is the one novel technique we examine.

²These abbreviations are used in the figures.

³Abbreviations are concatenated to indicate the complete technique. For example, HcSr indicates hill climbing using random swap mutation.

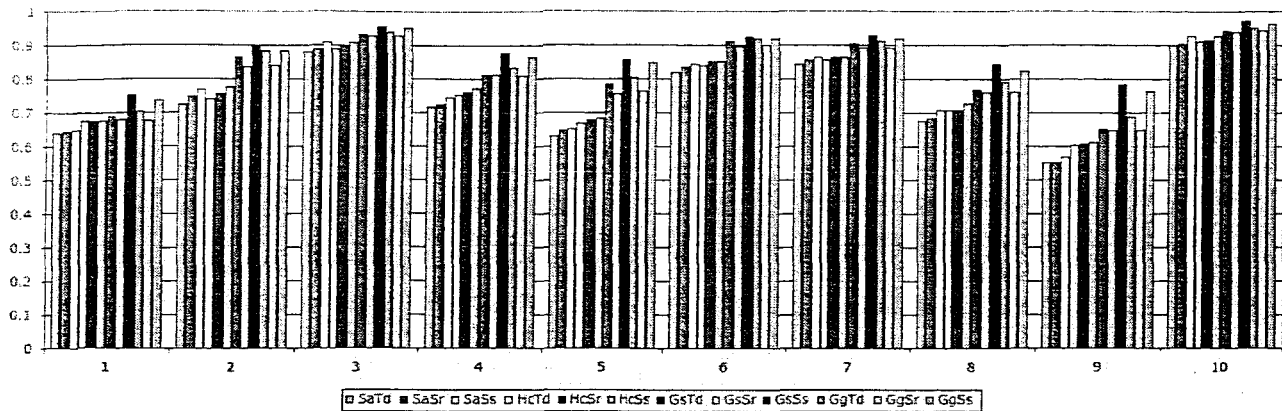


Figure 1: Relative fitness of all techniques on all problems normalized by ISAMP fitness. The vertical axis is the mean of the fitness divided by ISAMP fitness for the same problem. The mean is taken from the most fit schedules for each of 32 jobs for each technique. Lower values indicate better fitness. The horizontal axis is the technique. Color/shade indicates the problem. Sa = simulated annealing, Hc = hill climbing, Gs = steady state genetic algorithm, Gg = generational genetic algorithm, Td = temperature dependent swaps, Sr = random swaps, Ss = squeaky shifts. For example, on problem 9 simulated annealing with temperature dependent swap (SaTd) had a mean best fitness 55% of ISAMP on the same problem, whereas in problem 4 the fitness was 71% of ISAMP.

3. Squeaky shift (Ss). This implements squeaky wheel optimization. The mutator shifts 1-15 (randomly chosen) 'deserving' observations earlier in the permutation. Early in the permutation an observation is more likely to be scheduled since fewer other observations will have been scheduled to create constraints. Each observation to shift forward is chosen by a tournament of size 50, 100, 200, or 300 (chosen at random each time). The observation is always chosen from the last half of the permutation. The position-to-shift-in-front-of is chosen by a tournament of the same size (each time) and is guaranteed to be at a location at least half way towards the front of the permutation (starting at the 'deserving' observation). The observation most deserving to move earlier in the permutation is determined by the following characteristics (in order):

- (a) unscheduled rather than scheduled
- (b) higher priority
- (c) later in the permutation

The position-to-shift-in-front-of tournament looks for the opposite characteristics.

We tested a number of other mutation operators in preliminary experiments (Globus *et al.* 2003). The ones examined in this experiment performed the best.

In the case of the genetic algorithms, half of all children are created by mutation and the other half by crossover. Crossover combines two parents to create a child. The crossover operator is called position-based crossover (Syswerda & Palmucci 1991). Roughly half of the permutation positions are chosen at random (50% probability per position). The observations in these positions are copied from the father to the same permutation location in the child. The remaining observations

fill in the child's other permutation positions in the order they appear in the mother. For example:

```
mother:  5 4 3 2 1
father:  1 2 3 4 5
choose:  x   x x
child:   1 3 2 4 5
```

We also tested heuristic-biased stochastic sampling (HBSS) (Bresina 1996) with contention heuristics (Frank *et al.* 2002), an algorithm proposed for the EOS observation scheduling problem. This technique is not permutation based, it uses heuristics to repeatedly choose the next observation request to place in the timelines and which observation window to use. The contention data from which the heuristics are calculated must be updated as observations are scheduled, an expensive process. HBSS was hundreds of times slower than the evolution-based techniques, required far more memory, and produced very poor schedules.

Experiment

To find the best algorithm for the model problems we compared a total of thirteen techniques. These were ISAMP and every combination of four search techniques – hill climbing, simulated annealing, steady state GA, and generational GA – crossed with three mutation operators – 1-15 random swaps, 1-15 temperature dependent swaps, and 1-15 squeaky shifts. Thirty-two jobs with identical parameters (except the random number seed) were run for each algorithm. Each job generated approximately 100,000 schedules (the GA jobs generated slightly more). Most jobs ran in 2-3 hours on a modern Linux Pentium processor with plenty of memory (no swapping or paging to disk).

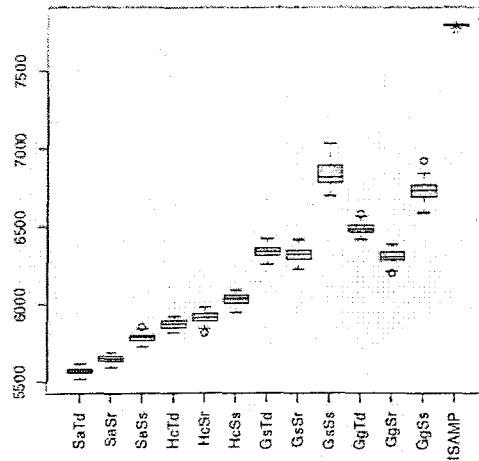


Figure 2: Fitness distribution comparison for all techniques on problem 4. The vertical axis is fitness (lower values are better), the horizontal axis technique. The boxes indicate the second and third quartiles. The line inside the box is the median and the whiskers are the extent of the data. Outliers are represented by small circles.

In any study of this kind it is always possible that the results would have been different if one algorithm or another had used a different set of parameters (population size, temperature schedule, hill-climbing restarts, etc.). The number of potential combination is literally astronomical and one must, in the interest finishing within the lifetime of the universe, chose some set to test. For the same reason, the reader's favorite technique may have been left out.

We spent considerable time in preliminary experiments searching for good GA parameters, and some time looking for the best restart schedule for hill climbing, but the simulated cooling schedule was the first we tried. The population size of 100 was chosen after many jobs with random population sizes were examined and the best results appeared around 100. Several restart regimes were examined for hill climbing but the differences were generally not statistically significant. The simulated annealing cooling schedule was set such that considerably less fit schedules could win near the beginning of search but by the end of search simulated annealing became almost pure hill climbing. From the beginning simulated annealing out-performed all other techniques so we never felt a need to improve the cooling schedule; this could only increase simulated annealing's lead. We also ran preliminary experiments to chose the number of mutation swaps and other parameters common to several techniques.

Most of the differences observed in the results were statistically significant by both t-test and ks-test, with confidence levels usually far above 99%. The raw data is available on request.

Results and Discussion

Figure 1 compares the algorithm's fitness performance for all problems, normalized by ISAMP's performance

so all results are in the range 0-1. The rising slope of the bars within each problem indicates that simulated annealing was best, followed by hill climbing. The genetic algorithms performed worst on all problems. The differences are not academic. The best techniques schedule hundreds more observations than the worst on most of the model problems.

Figure 2 compares fitness for all jobs and all techniques on problem 4. Results for the other nine problems are similar. Note that the range of fitness values within jobs running the same technique is very small. Thus, almost all of the differences are statistically significant by the t-test and ks-test. In most cases, the probability that the distributions are different is $>> 99\%$.

Simulated annealing performance benefits from a proof of optimality, at least for an infinitely slow cooling schedule. Hill climbing is vulnerable to local minima, but performs nearly as well as simulated annealing anyway, suggesting that there are local optima but their depth is not greatly different. Both hill climbing and simulated annealing outperform both variants of the genetic algorithm. This does not appear to be caused by poor choice of GA parameters. Rather, the EOS scheduling problem appears to favor exploitation over exploration.

The tight distribution of fitness values around the median also suggests that all jobs found the same minimum or that, if the fitness landscape is multi-modal as the Sa vs Hc results suggest, most minima must be about the same. Since both hill climbing and simulated annealing spend all of their time on a single individual and the GA must spread its search over a population, GA does less exploitation and loses. It is possible that a smaller population and/or larger tournament size could reduce this effect. The population size was, however, selected on the basis of preliminary experiments.

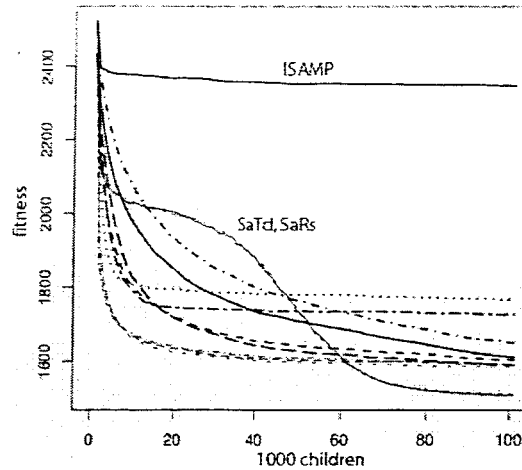


Figure 3: Evolutionary history on problem 1. The vertical axis is the median fitness of the best individual for 32 jobs for each technique. The horizontal axis is the number of schedules generated (children). The lines are the techniques. ISAMP, SaTd, and SaSr are labeled. Note that the best techniques (simulated annealing with variations of random swap) do poorly after only 20-50 thousand children, but are by far the best by 100,000 children.

Examining figures 1 and 2 carefully, we see that temperature dependent random swaps performs better than random swaps for simulated annealing and hill climbing and worse for steady state genetic algorithm and generational GA, although not by much. In units of ISAMP mean fitness, the differences were 0.01 for simulated annealing, 0.0067 for hill climbing, 0.01 for steady state GA, and 0.027 for generational GA averaged over all problems. Thus, temperature dependent swaps are a bad idea for the population based techniques but of some value on the others. These differences were usually, but not always, statistically significant.

The squeaky shift mutator performs worse than the other mutation operators, particularly for the genetic algorithms. Relative to temperature dependent swaps in units of ISAMP mean fitness: 0.024 of ISAMP for simulated annealing, 0.015 for hill climbing, 0.05 for steady state GA and 0.024 for generational GA. This suggests that squeaky shift is smart in the wrong way.

In preliminary experiments we also tried swapping, rather than shifting, observations. The shift operator performed the best, but still not as well as the random swap mutator. If random outperforms intelligent, then clearly intelligence is being poorly applied. We do not understand the dynamics of permutation-space scheduling in any fundamental way, and we don't even know if the dynamics are similar for different problems. Until a better understanding is reached, the random swap operators – with a optional decrease in the number of swaps as evolution proceeds – appear best.

Figure 3 shows the fitness evolution history for all techniques as a function of the number of children generated for problem 1. Note the rapid improvement for all techniques in the beginning, with ISAMP quickly leveling out.

Most techniques then show an elbow shape, rapid

improvement followed by a quick transition to very slow improvement. However, the best two techniques, simulated annealing with random swaps or temperature dependent swaps, have a different evolutionary shape. They actually perform worse than everything but ISAMP between 20,000 and 50,000 children. Thus, if we had run our experiment for 50,000 children we would have quite different results. All techniques have small slopes at 100,000 children so we do not expect the results to change with longer runs. The evolutionary curves for most other problems had a similar shape, although the best simulated annealing techniques were rarely worse than all others (except ISAMP) in the 20,000-50,000 children range.

Conclusions

We compared thirteen different permutation-space search techniques on ten realistically-sized EOS scheduling problems. Simulated annealing outperformed hill climbing which, in turn, greatly outperformed the genetic algorithm. Simple random swap mutation outperformed the more 'intelligent' squeaky mutation. Reducing the number of random swaps as evolution proceeds further improved performance for simulated annealing and hill climbing, but only slightly. Essentially the same results have been observed in preliminary experiments.

Future EOS scheduling applications, at least those choosing a permutation representation, should strongly consider simulated anneal with either random swap or temperature dependent swap mutation. These are simple to implement, fast, and appear to be superior for EOS observation scheduling.

An important follow-up to our work would be an equally thorough study of non-permutation methods;

those that search in the space of all possible schedules. We conjecture that the simplicity of local search in permutation-space (particularly the fact that we do not need to search in infeasible space) will lead permutation-based methods to dominate. However, this conjecture can only be evaluated by a head-to-head comparison of the best permutation-based and schedule-based search algorithms.

Acknowledgements

This work was funded by NASA's Computing, Information, & Communications Technology Program, Advanced Information Systems Technology Program (contract AIST-0042), and by the Intelligent Systems Program. Thanks to Greg Hornby and Bonnie Klein for reviewing this paper and to Jennifer Dungan, Jeremy Frank, Robert Morris and David Smith for many helpful discussions. Finally, thanks to the developers of the excellent Colt open source libraries for high performance scientific and technical computing in Java.

References

- Baluja, S. 1995. An empirical comparison of seven iterative and evolutionary function optimization heuristics. Technical Report CMU-CS-95-193, Carnegie Mellon University.
- Bensana, E.; Lemaitre, M.; and Verfaillie, G. 1999. Earth observation satellite management. *Constraints* 4(3):293-399.
- Bresina, J. 1996. Heuristic-biased stochastic sampling. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*.
- Crawford, J. M., and Baker, A. B. 1994. Experimental results on the application of satisfiability algorithms to scheduling problems. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*.
- Frank, J.; Jonsson, A.; Morris, R.; and Smith, D. 2002. Planning and scheduling for fleets of earth observing satellites. In *Proceedings of the 6th International Symposium on Artificial Intelligence, Robotics, Automation and Space 2002*.
- Globus, A.; Crawford, J.; Lohn, J.; and Morris, R. 2002. Scheduling earth observing fleets using evolutionary algorithms: Problem description and approach. In *Proceedings of the 3rd International NASA Workshop on Planning and Scheduling for Space*.
- Globus, A.; Crawford, J.; Lohn, J.; and Pryor, A. 2003. Scheduling earth observing satellites with evolutionary algorithms. In *Conference on Space Mission Challenges for Information Technology (SMC-IT)*.
- Holland, J. H. 1975. *Adaptation in Natural and Artificial Systems*. Oakland, CA: University of Michigan Press.
- Husbands, P. 1994. Genetic algorithms for scheduling. *AISR Quarterly* (89).
- Joslin, D. E., and Clements, D. P. 1999. Squeaky wheel optimization. *Journal of Artificial Intelligence Research* 10:353-373.
- Kirkpatrick, S.; Gelatt, C. D.; and AndVecchi, M. 1983. Optimization by simulated annealing. *Science* 220(4598):671-680.
- Lamaitre, M.; Verfaillie, G.; Frank, J.; Lachiver, J.; and Bataille, N. 2000. How to manage the new generation of agile earth observation satellites. In *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space*.
- Lamaitre, M.; Verfaillie, G.; and Bataille, N. 1998. Sharing the use of a satellite: an overview of methods. In *SpaceOps 1998*.
- Potin, P. 1998. End-to-end planning approach for earth observation mission exploitation. In *SpaceOps 1998*.
- Potter, W., and Gasch, J. 1998. A photo album of earth: Scheduling landsat 7 mission daily activities. In *SpaceOps 1998*.
- Sherwood, R.; Govindjee, A.; Yan, D.; Rabideau, G.; Chien, S.; and Fukunaga, A. 1998. Using aspen to automate eo-1 activity planning. In *Proceedings of the 1998 IEEE Aerospace Conference*.
- Syswerda, G., and Palmucci, J. 1991. The application of genetic algorithms to resource scheduling. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, 502-508.
- Watson, J.-P.; Barbulescu, L.; Howe, A. E.; and Whitley, D. 1999. Algorithm performance and problem structure for flow-shop scheduling. In *AAAI/IAAI*, 688-695.
- Wolfe, W. J., and Sorensen, S. E. 2000. Three scheduling algorithms applied to the earth observing systems domain. *Management Science* 46(1):148-168.